The AirTight Protocol for Mixed Criticality Wireless CPS

J. HARBIN, A. BURNS, R.I. DAVIS, L.S. INDRUSIAK, I. BATE, and D. GRIFFIN, Department of Computer Science, University of York, UK.

This paper describes the motivation, design, analysis and configuration of the criticality-aware multi-hop wireless communication protocol AirTight. Wireless communication has become a crucial part of the infrastructure of many cyber-physical applications. Many of these applications are real-time and also mixed-criticality, in that they have components/subsystems with different consequences of failure. Wireless communication is inevitably subject to levels of external interference. In this paper we represent this interference using a criticality-aware fault model; for each level of temporal interference in the fault model we guarantee the timing behaviour of the protocol (i.e. we guarantee that packet deadlines are satisfied for certain levels of criticality). Although a new protocol, AirTight is built upon existing standards such as IEEE 802.15.4. A prototype implementation and protocol-accurate simulator have been produced. This paper develops a series of schedulability analysis techniques for single-channel and multichannel wireless Cyber-Physical Systems (CPS). Heuristics are specified and evaluated as the starting point of design space exploration. Genetic algorithms are then defined and evaluated to assess their performance in developing schedule tables incorporating multichannel allocations in these systems.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; **Real-time systems**; • **Software and its engineering → Software verification**;

ACM Reference format:
J. Harbin, A. Burns, R.I. Davis, L.S. Indrusiak, I. Bate, and D. Griffin. 2019. The AirTight Protocol for Mixed Criticality Wireless CPS. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (January 2019), 28 pages.
DOI: 10.1145/3362987

1 INTRODUCTION

Many Cyber-Physical Systems (CPS) require some form of wireless communication and contain components/subsystems of different levels of criticality. AirTight [8] is a wireless protocol that is designed to meet the challenge of supporting mixed-criticality real-time traffic between computational nodes. In this paper we significantly expand the initial definition of AirTight [8] and consider how a complete system can be specified via heuristics and the application of genetic algorithms.

Mixed-criticality scheduling analysis [6, 38] allows resources, such as processor time or communication bandwidth, to be managed in a way that enhances schedulability while ensuring that the more critical/important work is protected. A focus on mixed criticality communication has led to the definition of criticality-aware protocols and analysis for Network-on-Chip (NoC) [7, 23] and CAN [5]; here we extend this work to cater for wireless communication. Unfortunately no existing complete protocol gives the right level of support for event- and time-based communications that have hard deadlines for packet delivery (see related work in Section 3). AirTight is a new protocol that is built upon the physical and MAC layers of IEEE 802.15.4, a standard for wireless personal area networks (WPANs), widely used as the basis of protocols such as ZigBee and WirelessHART.

With wireless communication, it is not realistic to only require that deadlines are met when there are no faults. Rather, as in other considerations of fault tolerance, we require that certain levels of performance are delivered when the likelihood and severity of fault(s) is bounded by what is referred to as a *fault model*. We assume that the physical layer of the protocol incorporates the usual methods of increasing resilience (for example spectrum spreading), and we therefore focus on faults that manifest themselves as unacknowledged frame transmissions at the MAC layer.

In this paper the analysis developed for AirTight allows deadlines to be guaranteed at various levels of service, corresponding to the severity of the fault model and the level of service required by the criticality assigned to the packet being guaranteed. Different fault models can be applied; the basic behaviour of AirTight is not dependent upon any particular fault model.

The first contribution of this paper is to present new analytical techniques for wireless Cyber-Physical Systems. Firstly, the analysis in our previous AirTight paper [8] is extended to support multi-hop flows by considering the summation of individual hop response times along the route. This improved analysis is referred to in this work as Single Channel Analysis (SCA). This removes the requirement for each hop of a multi-hop route to meet a particular individual deadline, as long as their final summation meets the flow deadline.

The second contribution is to consider multichannel scheduling. Multiple orthogonal channels allowing parallel non-interfering transmissions are a standard feature of the underlying physical layers upon which wireless CPS are built. This imposes a wider design space, such as determining an allocation of particular transmissions to channels. Also additional constraints are imposed, such as the requirement that peer nodes be correctly tuned to the necessary channel to hear their intended transmitter. In this paper multichannel analysis (MCA) is developed based on the concept of affinity sets [19] and the associated scheduling equations are presented and explained.

Our third contribution is to specify and evaluate optimisation processes based upon Genetic Algorithms for the construction of AirTight scheduling tables. We consider the automated generation of slot tables for synthetically generated networks, using genetic algorithm optimisation to refine a population originally generated using simple slot table heuristics.

In the next section we discuss the requirements for which AirTight was defined. We also give an overview of the protocol and define some necessary terms. In Section 3 we consider related work. Section 4 describes the fault model employed. Section 5 completes the description of the AirTight protocol, and its analysis is given in Sections 6 and 7. We specify heuristics as a starting point for the exploration of the design space of CPSs in Section 8, and give a numerical example in Section 9. We then define a genetic algorithm search process in Section 10. Experimental evaluation of the heuristics and GA search process is performed in Section 11. Finally conclusions are provided in Section 12.

2 REQUIREMENTS FOR, AND OVERVIEW OF, AIRTIGHT

We assume that the system consists of a distributed set of nodes that can each perform any combination of executing tasks, producing/consuming data from sensors/tasks, and relaying data packets to and from other nodes. There may be a range of communication media within the Cyber-Physical System; here we focus on the use of wireless technology. The required wireless network protocol is assumed to have the following properties (most of them inherited from the parent standard IEEE 802.15.4):

- Peer-to-peer packet-switching communication between tasks/nodes is the normal use of the network. Packets are sent from a node to the next as one or more *frames*. Each successful frame transmission is always acknowledged by the receiver through the transmission of a short ACK frame.

- Multi-hop routing is required due to the limited transmission range of each node, meaning that some packets are unable to be sent directly to their destination.
- Buffers exist on each node to store frames in transit (the size of the buffers required on each node can be determined during the off-line scheduling process).
- The nodes have their clocks synchronised so the maximum drift between any two clocks is at most $T_{error} \ll S$ (the slot length).
- Nodes have line power, so energy efficiency/battery life is not a limiting concern.
- Multiple frequency bands (channels) are available in IEEE 802.15.4 (up to 16 in the 2.4GHz band) and the standard is designed so that interference from one channel to another is negligible. A node can only use one channel at a time.
- Node communications are represented by two graphs: the *communications graph* and the *interference graph*:
  - The communications graph **C**: if there is an edge from $A \rightarrow B$ in **C**, then the two nodes can communicate directly. This is required to be a symmetric graph due to the necessity for an acknowledgement to be returned to the sender, so $A \rightarrow B$ implies $B \rightarrow A$.
  - The interference graph **I**: if there is an edge from $A \rightarrow B$ in **I**, then a transmission from A will prevent B from successfully receiving a frame from any node (other than A and then only if $A \rightarrow B$ is in **C**).

It is assumed that the packets to be communicated have tight timing constraints (i.e. deadlines). We also require that the system supports applications of different levels of criticality. We will, in this paper, assume just two criticality levels, high (HI) and low (LO). The main distinction between these levels is the number and duration of faults that they must tolerate (see Section 4).

The distributed system consists of N nodes (n_0 to n_{N-1}). Each node generates a set of packet flows (or flows), τ_i, defined by:

- Period, T_i; the minimum time between packets.
- Capacity, C_i; the packet's maximum size.
- Criticality Level, L_i; a static parameter[1] of the flow.
- Deadline, D_i; assumed initially to be no greater than T_i.
- Destination, Des_i; packets are assumed to be peer-to-peer so there is a single destination for each flow.
- Source, Src_i; there is an implicit source for each flow.

As part of the configuration of local scheduling, unique local priorities are assigned to each of the flows transmitted by a node. This is done for every node.

We do not assume that the flows are purely periodic. This implies that there must be some form of run-time scheduling. However, we do not expect that centralised access control, or token passing protocols can deliver the performance required by a modern Cyber-Physical System. Any protocol that requires significant overhead to agree on the next packet to send is unlikely to meet strict timing requirements. The alternative of a fully table-driven time-triggered protocol lacks the flexibility needed to support event-triggered and adaptive applications.

AirTight is designed to balance efficiency and flexibility. At the system level, its media access control is table-driven, but at the node level it uses criticality-aware priority-based frame scheduling. The protocol is based on slot tables which define a repeating cycle of activities for each node – either transmission or reception on a given channel, or null meaning no usage.

[1]The criticality level of a flow is inherited from the criticality level of the application(s) that it forms part of. This is a static property, not to be confused with the criticality mode of a node or the system, which is a dynamic property.

At each node, local scheduling decisions are made to manage the use of the node's slot allocation. We employ a fixed-priority scheme (although this is not fundamental to AirTight). A set of FIFO queues (buffers), one per priority level, are used to hold the frames that need to be transmitted. Each flow has a unique priority at each node and hence a specific buffer that it can use. The frames from the same flow are stored in the buffer in FIFO order. Whenever the node has a transmission slot available, it transmits the first frame in the highest priority non-empty buffer. If an ACK is received, then the frame is removed from the buffer; if no ACK is received, then the frame remains in the buffer and is a candidate for re-transmission when the next available slot for that node becomes available.

AirTight is thus a two level protocol. A collection of slot tables (one per node) defines the usage of the wireless media. Each slot in a table defines whether the node can transmit in that slot (and on which channel if more than one channel is used), or whether it should listen in that slot (and on which channel), or whether it is off-duty. The collection of tables reflects the properties of the communication and interference graphs. So, within the same channel, two nodes may, in effect, be allocated the same slot if the interference graph fulfils specific properties with regards to senders and receivers [18].

The fundamental time unit of AirTight is the duration (S) of a slot – the time it takes to communicate a single frame of data and receive an ACK for that frame. All parameters of the application, the communication media and the environment (e.g. T_i, C_i, D_i, table length, fault models, etc.) are expressed as an integer number of slot times. We assume that clock drift is insignificant when compare to the slot duration. (In our prototype implementation [8] a slot length of 10ms has been achieved. This is the slot length in most WirelessHART implementations [17]).

Note that in contrast to mixed-criticality scheduling of processors where each task can have a LO-criticality and a (larger) HI-criticality worst-case execution time estimate, in the context of AirTight, we assume that a packet's maximum size C_i is known and fixed. Mixed criticality behaviour of the system is instead due to variability in the level of interference that must be tolerated by packet flows of different criticality levels, and hence the differing number of extra transmission slots that may be required to ensure correct transmission under an appropriate fault model - see Section 4.

2.1 Example of the Basic AirTight Protocol

Below, we provide a simple example, to illustrate the basics of the AirTight protocol. (The protocol is described in detail in Section 5). Figure 1 shows three nodes n_0, n_1, and n_2 and their connectivity.

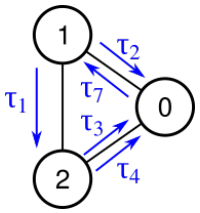


Fig. 1. Diagram of the nodes and communicating packet flows (basic example)

Also illustrated are five packet flows labeled τ_1 to τ_4 and τ_7 (this is part of a larger example used later). For simplicity, here we assume that there is only one channel available for transmissions and that all packets consist of one frame, except τ_1 which has two frames. Further, we assume that on node n_1, flow τ_2 has higher priority than τ_1 and on node n_2, flow τ_4 has higher priority than

τ_3. Finally, the same slot table is used by all three nodes and consists of a repeating cycle $\{1, 0, 2\}$. Flows τ_1, τ_3, τ_4 and τ_7 are initially ready to be transmitted and flow τ_2 becomes ready after 3 slot times ($t = 3$).

The AirTight protocol results in the following dynamic schedule. (Note "X" indicates a failed transmission, and "-" no transmission).

Time slot	0	1	2	3	4	5	6	7	8	9
Node eligable to transmit	1	0	2	1	0	2	1	0	2	1
Flow transmitted	1	7	4	X	-	3	2	-	-	1

At time $t = 0$, node n_1 is permitted to transmit, with the other two nodes listening on the channel. On node n_1, only flow τ_1 is ready for transmission, its first frame is therefore sent and acknowledged. At time $t = 1$, node n_0 is permitted to transmit, it sends the only frame of τ_7. At time $t = 2$, node n_2 is permitted to transmit. On this node, two flows are ready for transmission, τ_4 and τ_3, as τ_4 has higher priority its single frame is transmitted. At time $t = 3$ node n_1 is again permitted to transmit. Now both flows τ_1 (1 frame remaining) and τ_2 are ready. As flow τ_2 has the higher priority, its frame is transmitted. However, this transmission is not acknowledged due to some assumed interference. The frame of τ_2 therefore remains in the buffer, to be re-transmitted later. The dynamic schedule from this point is as follows. At time $t = 4$, there is no transmission, since node n_0 has this transmission slot but no ready packets. At time $t = 5$, node n_2 transmits the single frame of packet flow τ_3. At time $t = 6$, node n_1 succeeds in transmitting the frame of τ_2, leaving only the second frame of τ_1, which it is able to transmit at time $t = 9$.

This simple example serves to illustrate the two level protocol of AirTight: static global scheduling (use of slot tables) and dynamic local scheduling, based on priorities. This enables the protocol to react to higher priority flows becoming ready, and re-transmission requirements caused by interference.

3 RELATED WORK

In this section we consider wireless protocols that have been designed to give time predictable behaviour, and protocols that take into account mixed criticality. For information on more general purpose protocols the interested reader may follow the links from [31].

3.1 General Real-Time Protocols

WirelessHART [17] was developed as an extension of the HART protocol [16] designed for wired communications in industrial automation and process control scenarios. The WirelessHART protocol extensions were intended to allow mobile devices to attain the capabilities of HART networks. WirelessHART employs a time-division multiple access (TDMA) based MAC layer, with multi-hop routes centrally planned and allocated by a sink which operates as a gateway between the wireless network and external access network. A similar TDMA approach is advocated by FireFly [29, 32].

One notable aspect of WirelessHART is the avoidance of spatial reuse throughout the network, with only one simultaneous transmission allowed at any time. Although frequency reuse is permitted through the simultaneous use of multiple channels, simultaneous transmissions on any one channel are disallowed [35]. This avoids the problem of detecting interference patterns and allows the network to be stabilised, but limits scalability and restricts the viable range of the network. In addition, alternative routes are required in WirelessHART for redundant data transmission. By comparison, our current work focuses entirely on avoiding faults via temporal retransmission;

that is, holding transmissions within buffers and making repeated transmissions of any failed transmission after a delay.

Saifullah [33] presents several approaches for scheduling of multi-hop routing-centric wireless networks built upon WirelessHART. Saifullah establishes that overall the wireless scheduling problem is NP-hard, and then provides heuristics to simplify the scheduling decisions. The first of these, Conflict-Free Least-Laxity First (CLLF) [35] schedules according to the laxity of the transmissions, that is the time remaining until their deadlines. However, it guides the scheduling process by focusing on conflicts, scheduling first in hotspot areas in which conflicts are likely (e.g. around the sink and congested frequently used devices). Flow set evaluation results show that CLLF is several orders of magnitude faster in determining schedulability than exhaustive search, although run time does increase with increased routing diversity. Saifullah et al. also present an end-to-end delay analysis [36] for an application with fixed priority flows, and extend this work to the case of graph routing in [34].

A very different approach for broadcast wireless communication is provided via Glossy [14], which attempts to simultaneously achieve time synchronisation as well as error tolerant communication via collaborative flooding of data packets. If time synchronisation is sufficiently tight, multiple nodes are able to receive packets and rebroadcast them simultaneously. Glossy therefore uses spatial diversity to compensate for any localised faults; it has been successfully used as a primitive layer to build network services such as LWB [13] and Blink [44].

3.2 Mixed Criticality Protocols and Applications

Several papers have focused on mixed criticality in WSNs and CPS. Alemayehu et al. [1] consider mixed-criticality used for video transmission in a multimedia sensor network. The paper uses different criticality levels for different resolution video frames, to provide graceful degradation under errors by providing a low resolution alternative. They focus on changing criticality in response to a reduction in overall network bandwidth, and can discard data in response to criticality and priority to improve overall performance. An interesting contrast to our work is that the paper did not use TDMA and a pure wireless topology but a hybrid CSMA/CA over 802.11, yet still achieved end-to-end response time reductions.

Shen et al. [37] present the PriorityMAC protocol. The paper uses a concept of priority levels, but their definition overlaps with criticality – the importance of reliable delivery from the application perspective. It defines four traffic types and requires windows at the start of every application message for the two highest priority traffic levels to reserve capacity. Nodes must listen to these windows and detect them as clear if they wish to transmit the two highest priority traffic types. This allows dynamic adjustments in priority by allowing nodes with higher priority traffic to use slots previously assigned to other nodes, but at the cost of channel capacity reductions. Moreover, their analysis and evaluation only considers average-case latencies and does not provide a worst-case guarantee.

Jin et al. [24] considers delay analysis for WirelessHART networks supporting mixed criticality under fixed priorities. An interesting aspect of their model is that they assume a global network criticality level, and a broadcast mechanism to signal a criticality mode change similar to WPMC-FLOOD [23] within a network-on-chip (NoC). However, they do not specify any low level router behaviour to achieve the change, merely assuming the change takes a maximum defined time. It does not appear to use criticality monotonic arbitration, instead assuming that the LO-criticality flows are dropped rather than buffered, due to the low memory capacity of the devices.

StealRM [25] is an alternative protocol which provides redundancy and is employed for HI-criticality flows, with the transmissions of HI-criticality packets along two duplicate routes to

protect against interference or damage to one copy. The algorithm centrally establishes schedules for both the LO and HI-criticality flows, but allows nodes to make the distributed decisions for transmission of a HI-criticality flow in place of a LO-criticality flow when required. This is done by means of a clear channel assessment performed by the radio before making every LO-criticality transmission, which ensures that interference is avoided. Therefore, LO-criticality packets may be destroyed when HI-criticality packets require the resource, referred to as slot stealing. This approach is similar to the approach of Shen et al. with PriorityMAC [37], since it allows nodes to request additional capacity dynamically.

Dimopoulos et al. [12] consider mixed criticality systems, specifically for smart building infrastructure. One interesting aspect they mention is the potential use of SDRs (software defined radios) in the implementation of CPS, in order to present more adaptability in the behaviour and protocols employed. In nodes with more resources than conventional WSN motes this may be a viable solution. They also consider mixed criticality in wireless systems to require levels of autonomous management in different regions, contrasting with the implicitly centralised management and routing-centric designs required for WirelessHART (and assumed in the extensive scheduling studies performed in [33]).

Xia [41] extends the slot stealing mechanism as a way to handle additional emergency traffic in industrial CPS, potentially including alternative routes for emergency traffic. Although, since it does not inform the entire network of the emergency via a functional mode change, it cannot free up additional processing to handle the fault/emergency by stopping unnecessary LO-criticality tasks. Xia [40] considers EDF scheduling in a mixed criticality CPS, incorporating different routing strategies per criticality level. Graph routing is activated on a mode change in order to enhance the path diversity, and analytical techniques consider the demand bounds at intermediate nodes. The evaluation considers schedulability parameters but does not consider the impact of temporal faults upon the system. Recently, multiple-input multiple-output (MIMO) antenna technology has also been applied in mixed-criticality CPS [39] in order to provide additional capacity for HI-criticality flows at bottlenecks. Although shown to improve schedulability performance, the requirements for some heterogeneous hardware (MIMO) nodes would be an additional installation cost.

3.3 Comparison of AirTight To Reviewed Protocols

Compared to the WirelessHART protocol [17], AirTight provides simplicity under increased fault conditions due to the ability for the nodes to make entirely local decisions, without requiring global scheduling of all network flow transmissions. The same applies to scheduling techniques such as Saifullah's CLLF [35] which are built upon WirelessHART. Protocols built upon Glossy [14], such as LWB [13] and Blink [44] require very tight time synchronization for physical layer transmissions to the level of inserting NOPs in system code to ensure correct timing, adding implementation difficulties compared to AirTight. Also, since every packet has to be flooded through the network, utilisation would be much lower than in a highly peer-to-peer system incorporating AirTight.

Jin [24] incorporates a network broadcast criticality change, which requires a global change. Other works [25, 37] require specific radio hardware support, in the form of channel clear assessment or listening detection in particular slots. Compared to the units evaluated in [12], AirTight does not require software defined radios and is capable of functioning upon commodity devices with fixed radio transceivers. Xia et al. [41] does not incorporate mode changes so cannot free up additional processing to handle the fault situation by stopping LO transmissions; [40] considers schedulability parameters but does not consider the impact of long-lasting temporal faults. Compared to [39] which requires MIMO antennas and [12] which relies upon SDRs, AirTight is capable of functioning upon existing commodity devices with fixed radio transceivers.

In summary, by comparison with the above approaches, AirTight is the first mixed criticality wireless protocol that incorporates completely local scheduling decisions, and which delivers time-bounded performance that is sensitive to whatever fault model is deemed appropriate for the system under consideration.

4 FAULT MODEL

A wireless network, even in a protected domain, will suffer interference that will result in some packets being corrupted. A predictable network can only be derived and analysed if there is a bound on the level of interference suffered by each node in the system. This bound is usually expressed as a *fault model*. If the level of interference is no worse than that implied by the fault model then temporal guarantees can be made. The quality of the fault model can itself be modelled using a probabilistic estimate of the likelihood of exceeding a given fault severity during, say, an hour of operation [9]. With mixed-criticality systems the required quality will vary with criticality; so for a LO-criticality transmission the fault model may bound the number of deadlines misses to be no more than 1 in 1000, for a HI-criticality transmission this number may be extended to 1 in 1,000,000.

In general a node will suffer interference from a number of different sources. Each source will produce a pattern of interference. Moreover, in a geographically distributed network each node will experience different levels of interference from different sources. To model a particular node's (n_k) level of interference we need a *fault load function*, F_k. This function, when given an interval of duration t, will return the level of interference assumed by the fault model for this node at criticality level, L; i.e. the function is defined as $F_k(L, t)$. As the basic time unit in the analysis model is the duration, S, of a single slot, both t and F_k are represented as an integer multiple of S.

We note that the fault model is always assumed to be more severe for HI-criticality packets than for LO-criticality packets. Hence we require that:

$$\forall t, \forall n_k : \; F_k(HI, t) \geq F_k(LO, t)$$

The function F_k can be decomposed into a combination of fault load functions (f_k) for each of the w sources of interference:

$$F_k(L, t) \;=\; G_L\{f_k^1(L, t), f_k^2(L, t), \ldots, f_k^w(L, t)\}$$

where G_L is a criticality-specific application-defined means of combining the different sources of interference.

So, for example, for LO-criticality packets G_{LO} may be defined to be the MAX operator, and hence at this criticality level the node is assumed to only suffer interference from one source at a time – but the maximum possible single level is used to define F_k. For HI-criticality packets G_{HI} may be defined to be the SUM operator, and hence all sources are assumed to contribute their maximum levels – a situation that may be impossible as interference is not cumulative.

The most straightforward way of representing a single source of interference f_k is via a duration and a frequency. So a single corruption could cause a blackout for duration $b(L)$, with the minimum time between faults: $T^b(L)$. Note these parameters are functions of criticality level and are measured in units of S. A single source of interference could, for instance, be modelled by $b(LO) = 4$, $T^b(LO) = 100$, $b(HI) = 6$, $T^b(HI) = 80$: for LO-criticality transmissions this interference is assumed to last up to 4 units of time and repeat every 100 units; but for HI-criticality transmissions a more severe view is taken, the blackout can last 6 units of time and repeat every 80 units.

For an actual deployment of AirTight various signal processing schemes (for example Fourier Analysis) are available that allow the overall interference experienced at a node to be decomposed into component sources defined by these two parameters. These are statistical methods and hence

the parameters derived are a functions of the level of confidence required. Higher levels of criticality will require higher levels of confidence and hence more conservative parameters will be obtained.

In this paper we do not address further this analysis of actual interference, rather we assume that by the time an implementation requires analysis the necessary fault load functions have been obtained. All that the analysis requires is that $F_k(L, t)$ is defined for all time intervals t, all nodes n_k and for each criticality level in the system, and that $F_k(L, t)$ is a monotonically non-decreasing function of t. (Note that the analysis for the AirTight protocol developed in this paper (Sections 6 and 7) considers dual-criticality systems). In future, it could potentially be extended to multiple criticality levels, as has been done with the analysis for tasks [15].

5 THE AIRTIGHT PROTOCOL

In general a wireless network can be characterised by a number of properties:

- Single-hop or multi-hop (i.e. is the communications graph fully connected?).
- Single-domain or multi-domain (i.e. is the interference graph fully connected?).
- Single-channel or multichannel.

In this first journal paper on AirTight we focus on multi-hop, single-domain, multichannel networks. The extension to multi-domain does not introduce any fundamental issues, but requires a more complicated construction for the slot tables.

The protocol has three main phases:

(1) The construction of the slot table. This is derived from the requirements of all the packet flows on all the nodes. The table is communicated to all nodes during system initialisation.
(2) The run-time local scheduling of flows. Each node will, independently, make use of the slots allocated to it. This will take account of priorities, errors, and re-transmissions.
(3) An adaptive system will, over time, look to modify the slot table – for example there could be free slots that nodes compete for, or unused slots that are reallocated, or potentially the complete table could change due to a system mode change.

Analysis is used on each node to check for packet flow schedulability. This requires knowledge of the slot table; however, the structure of the slot table is itself a function of the schedulability of all nodes. In Section 6 we first derive analysis, assuming a known slot table, and then show how the slot table can be constructed with a simple heuristic. We then make use of a search-based algorithm (a GA) to: construct (near-optimal) slot table layouts, cover all required routing decisions, and cater for multichannel systems. The third phase (adaptation) is left for future work.

A schedulable AirTight network behaves as follows:

- If there are no faults experienced by the system then all packets will meet their deadlines.
- If the faults experienced by the system are no worse than that implied by the LO-criticality fault model then all packets will meet their deadlines.
- If the faults experienced by the system are no worse than that implied by the HI-criticality fault model then all HI-criticality packets will meet their deadlines.
- If the faults experienced by the system are worse than that implied by the HI-criticality fault model then each node will apply a best-effort approach. The faults are deemed to be beyond the level at which guarantees can be provided.

Following the behaviour of mixed criticality task scheduling [4], three modes of operation are defined. Each node is, independently, in either LO-criticality, HI-criticality or Best-Effort mode. In the LO-criticality mode all of the node's packets are sent and they are delivered by their deadlines. If the LO-criticality fault model is exceeded, then the node moves to HI-criticality mode. In this mode, LO-criticality packets are abandoned (or moved to local background priority); however,

all HI-criticality packets are still delivered by their deadlines. If the HI-criticality fault model is exceeded, then the node moves to Best-Effort mode. At any time that the output buffers of the node are empty the node can return to LO-criticality mode. (This is equivalent to the return to LO-criticality mode on an idle-tick in task scheduling).

It is possible to consider an alternative approach for HI criticality traffic in addition to best effort. In this alternative model, a given number of faults upon particular ultra-HI criticality traffic flows can trigger a network-wide change to UH (ultra-HI criticality mode). In this approach, an alternative delivery protocol is used for this UH data to give an increased probability of delivery to the destination. This idea has been explored in [21] but is not considered further in this paper.

Usually with distributed systems it is assumed that the packets inherit significant release jitter from the variability in the completion times of the tasks that generate them. This jitter can then be factored into the response-time analysis [3]. Here we apply a protocol that eliminates release jitter while not extending the worst-case overall (i.e. end-to-end) response-time [11]. Release jitter is eliminated by the following protocol which is applied to all frames of all packets. For clarity we describe its application to a single frame f of a single packet flow τ_i. The time q when frame f of the first packet of packet flow τ_i is received by node n_k is recorded. When at a later time t the node receives frame f of the next packet of the same packet flow, then: (i) if $t \geq q + T_i$, then the frame is immediately eligible for onward transmission by n_k and q is set to t; (ii) if $t < q + T_i$, then the frame is held (i.e. delayed) and is not eligible for further transmission along its route by n_k until time $q + T_i$ is reached. At that point q is set to $q + T_i$. The same process is repeated for subsequent frames of all packets of that flow. This protocol also applies to frames "received" by the source node from the sending task, with the initial maximum jitter due to the sending task (i.e. its worst-case response time) deducted from the end-to-end deadline. The effect of the protocol is to eliminate the interference effects of jitter, and thereby improve schedulability.

6 BASIC AIRTIGHT ANALYSIS FOR ONE CHANNEL

The starting point for the analysis of a complete system is a set of single-hop packet flows with Destination and Source nodes directly linked in the communications graph. In other words, all routing requirements have been met by the addition of intermediate flows passing between connected nodes (we revisit this issue in Section 8.3). We also assume local priorities have been assigned to all packet flows (an optimal assignment can be obtained by applying Audsley's algorithm [2]).

For any particular phase of the analysis the slot table is known. It has duration T_{SL} (measured in slots). Each node has one or more slots within the table; let this allocation be represented by a_i. We have (note, as before, N is the number of nodes in the system):

$$\sum_{j \in N} a_j = T_{SL}$$

The required analysis is obtained from adapting three schemes/notions:

- Modelling the impact of faults by a fault load function($F_k(L, t)$), which gives the maximum number of failed slots for criticality level L in time t for node n_k.
- Basic fixed-priority analysis for mixed-criticality task scheduling – using the AMC approach [4].
- Modelling the supply function ($S(X)$), the maximum time which the slot table can take to supply X slots to the node.

6.1 AMC Analysis for AirTight

AMC analysis was originally devised for a collection of tasks and exploits the fact that the load on the system is lighter during the LO-criticality mode, and the fact that LO-criticality tasks are dropped once the system transitions to the HI-criticality mode. With task scheduling the load is less in the LO-criticality mode as tasks have smaller worst-case execution time estimates in that mode [38]. When analysing packet flows, this is not the case (although the model could easily be extended to include this). Rather it is the fault load that is lower in the LO-criticality mode. This allows us to define response-time analysis for each packet flow, τ_i on node n_k. In the LO-criticality mode:

$$R_i(LO) \;=\; C_i \;+ F_k(LO, R_i(LO)) \;+ \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j \tag{1}$$

where **hp(i)** is the set of all local (i.e. also transmitted by node n_k on part of their route) flows with priority higher than that of τ_i. In the HI-criticality mode:

$$R_i(HI) \;=\; C_i \;+\; F_k(HI, R_i(HI)) \;+ \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j + \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k \tag{2}$$

where **hpH(i)** is the set of local HI-criticality packet flows with priority higher than that of flow τ_i; and **hpL(i)** is the set of local LO-criticality packet flows with priority higher than that of flow τ_i. Note $R_i(HI)$ is only defined for packet flows of HI-criticality. (Note all quantities in the above equations, including $R_i(LO)$ and $R_i(HI)$ are measured in units of the slot length S).

6.2 Sufficient Analysis for AirTight

The above analysis assumes that the required resources (the slots of the slot table) are always available for the node under investigation. This is a valid assumption for tasks executing on a single processor, since the processor is always available. With AirTight the slots are not as readily available. Indeed, as few as one in T_{SL} slots may be all that is available for the node under investigation.

We therefore represent the availability of slots as a supply function: $S_k(X)$, defined as follows: Given a slot table of length T_{SL} indicating the node that is permitted to transmit in each slot, the *supply function* $S_k(X)$ for node n_k returns the maximum number of slots that could elapse in a repeating static cycle of the table (starting at any point) before X slots allocated to node n_k occur, and hence node n_k could transmit X frames.

Equation (1) is now split into two parts:

$$X \;=\; C_i \;+ F_k(LO, S_k(X)) \;+ \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{S_k(X)}{T_j} \right\rceil C_j \tag{3}$$

with

$$R_i(LO) = S_k(X) \tag{4}$$

The equations are solved via fixed point iteration in the usual way, starting with an initial value of X of C_i. Iteration continues until either the value of X converges, in which case $R_i(LO)$ gives the worse-case response-time, or $R_i(LO)$ exceeds D_i, in which case the packet flow is not schedulable.

Similarly equation (2) becomes

$$X \;=\; C_i \;+\; F_k(HI, S_k(X)) \;+ \sum_{\tau_j \in \mathbf{hpH}(i)} \left\lceil \frac{S_k(X)}{T_j} \right\rceil C_j + \sum_{\tau_k \in \mathbf{hpL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k \tag{5}$$

with

$$R_i(HI) = S_k(X) \tag{6}$$

An example of the application of this analysis is given in Section 9.

A number of different formulas for $S_k(X)$ are possible. When node n_k has only one slot in the table ($a_k = 1$) then it must be assumed that the worst-case possible phasing between this slot and the packet flow under consideration occurs. This implies that a frame of the packet flow arrives just after the slot has been assigned to a lower priority packet, or indeed a null or background packet is assigned. Hence there is a 'blocking time' of 1 slot. Minor clock drift is also accommodated within this blocking term.

If no internal structure for the table is known then a sufficient model for the supply function is

$$S_k(X) = 1 + \left\lceil \frac{X}{a_k} \right\rceil T_{SL} \tag{7}$$

So, for example, if the table is of length of six and a node has one slot within the table, then the supply function returns 7 for $S(1)$, 13 for $S(2)$ and so on. Similarly if the node has two slots in the table, then its conservative supply function is $S(1) = S(2) = 7$, $S(3) = S(4) = 13$. If the internal structure of the table is known, then a less pessimistic estimate is possible. For example, the two slots cannot both come at the end of the table, so an improvement is $S(1) = 6$, $S(2) = 7$, $S(3) = 12$, $S(4) = 13$ etc. Moreover, if the table is known to have allocated the two slots to positions (1 and 4, or 2 and 5, or 3 and 6) then the supply function becomes: $S(1) = 4$, $S(2) = 7$, $S(3) = 10$ etc. This latter supply function dominates the more pessimistic ones, since it provides the same number of available slots in the same or less time, for any number of required slots.

7 MULTICHANNEL ANALYSIS

7.1 Affinity Set Basics and Structure

In a single-channel, single-domain network in which the interference graph is complete, only one transmission node can be active during a slot. Increasing the number of available channels clearly increases the capacity of the network, by permitting simultaneous transmissions using the additional channels available. There is another mechanism by which adding multiple channels can improve the overall performance of the algorithm, however. This involves partitioning the flows within the flowset so that each is restricted to transmission on a specific subset of channels. This approach is inspired by [19]. An affinity set is a mapping which given a flow τ_i returns a non-empty set of channels $\mathbf{Q}$ which the flow may use.

The key benefit of this is to reduce the size of the sets of higher priority flows using the same channels that need to be considered during analysis. However, the multichannel affinity sets may in some cases reduce performance. If a node has to send its transmitted flows on different channels, then it will require additional transmission slots on these channels (together with its receivers having to listen on these channels). Therefore, we will consider the optimisation of these affinity sets during our experimental work.

This concept of channel affinity also involves modifying the AirTight protocol so its transmission decisions for a slot assigned to a given channel Y, choose the highest priority flow that has affinity for channel Y. Of course, when a node makes a mode change, only a HI criticality packet with suitable affinity can be considered for transmission.

7.2 Multichannel Analysis (MCA)

In the multichannel analysis, fixed point iteration is used in a similar way to the SCA case. A key difference in considering the slot allocations is that in MCA a distinct supply function $SM_k(X, \mathbf{Q})$ is used. This supply function is defined as follows. Given a slot table of length T_{SL} indicating the sets of node-channel pairs that are permitted to transmit in each slot, the *supply function* $SM_k(X, \mathbf{Q})$

for node n_k, and a set of channels $\mathbf{Q}$, returns the maximum number of slots that could elapse in a repeating static cycle of the table (starting at any point) before X slots occur where node n_k is permitted to transmit on some channel in $\mathbf{Q}$.

A further difference is that only the higher priority flows that share a common channel in their affinity sets need be considered. Therefore, modified sets **hpA**, **hpAH** and **hpAL** are used in the analysis: **hpA(i)** is the set of all local flows with priority higher than that of τ_i sharing any common channel in their affinity sets with flow τ_i. **hpAH(i)** is the set of local HI-criticality flows with priority higher than that of flow τ_i that share any common channel in their affinity sets with flow τ_i. Similarly, **hpAL(i)** is the set of local LO-criticality flows with priority higher than that of flow τ_i that share any common channel in their affinity sets with flow τ_i. The response-time analysis becomes:

$$X = C_i + F_k(LO, SM_k(X, \mathbf{Q})) + \sum_{j \in \mathbf{hpA}(i)} \left\lceil \frac{SM_k(X, \mathbf{Q})}{T_j} \right\rceil C_j \tag{8}$$

$$R_i(LO) = SM_k(X, \mathbf{Q}) \tag{9}$$

$$\begin{aligned} X = C_i + F_k(HI, SM_k(X, \mathbf{Q})) + \sum_{\tau_j \in \mathbf{hpAH}(i)} \left\lceil \frac{SM_k(X, \mathbf{Q})}{T_j} \right\rceil C_j \\ + \sum_{\tau_k \in \mathbf{hpAL}(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k \end{aligned} \tag{10}$$

$$R_i(HI) = SM_k(X, \mathbf{Q}) \tag{11}$$

8 HEURISTICS FOR CPS SCENARIO SETUP

In order to design a system using the AirTight protocol, it is necessary to specify slot tables, channel affinity sets (if using a multichannel network) and routes (if needing multihop routing). Although this design space can be explored with search and optimisation techniques, it is better to have constructive heuristics as a starting point. In this section we first define simple heuristics for generating slot tables, multichannel affinity sets, and routes in a flowset incorporating multihop routing. This step is later used as a basis for evolutionary enhancement during a GA search process.

8.1 Slot Table Heuristic

Consider a flowset with multihop routes defined for its flows. The route for flow τ_i consists of H hops from n_{src} to n_{des}, with n_{des} being the destination node that only receives and does not re-transmit packets of the flow. The utilisation load that flow τ_i imposes on all of its transmitting nodes $n \in \{n_{src}, \ldots n_{des}\}$ can be determined from the flow's capacity (maximum packet size) and its period.

$$U_i = \frac{C_i}{T_i} \tag{12}$$

The total utilisation for node n_k, (UT_k) is the sum of all the utilisation values for all flows transmitted via the node.

Now it is necessary to convert UT_k into a slot count which can be used in the analytical techniques, i.e. substituted into the value of a_k in equation (7). This is done by rank-ordering all non-zero node utilisations and converting them into an integer slot number via the following relationship. The highest 25% of utilisations receive 3 slots, the next 25% receive 2 slots and final 50% receive 1 slot. Of course, nodes that do not engage in transmission (those with zero utilisations) receive zero slots.

8.2 Affinity Sets Heuristic

When performing multichannel scheduling, a simple approach for specifying initial channel affinity sets (as specified in Section 7.1) is to assign each flow to a fixed channel throughout its route. This assignment is performed in a round-robin fashion. For example, flow τ_i is assigned affinity to channel $i \ mod \ K$, where K is the number of channels. For example, with 3 channels, flows 1, 4, and 7 are assigned to channel 1. This serves to separate out the interference sets and thereby reduce the interference between flows.

8.3 Routing Heuristic

In general routing must be addressed as part of the mapping of an application to the available hardware, since it influences how intermediate nodes need to forward traffic, and therefore the loads experienced at these nodes. Several approaches have been used for route determination in systems derived from WirelessHART, which generally incorporate graph routing to/from a sink node; for example Han graph routing [20], Zhang graph routing [42], and Q-learning route construction [26]. Here we, initially take a simpler approach, since we are assuming a static industrial application with peer to peer traffic, in which the topology remains constant and routes can be pre-programmed. In Section 10 we describe in detail the use of a genetic algorithm to construct the tables. However, unlike in the Q-learning routing scheme just mentioned, the GA approach is applied prior to deployment, and only the best multihop route for each flow resulting from the optimisation process is used in the deployed system The interactions which come as a result of varying the routing and scheduling will be considered in future work. Hence:

(1) The shortest route between Source and Destination is chosen (an arbitrary choice is made if there is more than one route with the same length).
(2) The deadline of the packet is partitioned between each hop of the route, and local priorities are assigned to the flows transmitted by each node by applying Audsley's algorithm [2].
(3) Response times are computed for each hop and summed to obtain the end-to-end response-time that is then compared with the end-to-end deadline.
(4) Initially the packet deadline is divided equally between the hops, if any hop is unschedulable (i.e. $R_i > D_i$) its deadline is increased (to R_i, but not exceeding T_i) (while others are decreased when $R_i < D_i$).

Clearly this is a non-optimal approach, although the heuristic does account for 'busy' nodes by allowing them to have more slots in the table.

9 ILLUSTRATIVE EXAMPLE

9.1 An Avionics Use-case

A good example of the potential deployment of a wireless communication media is within an aircraft engine for the purposes of active health monitoring [43]. Figure 2 shows the communication graph for a 25-node wireless network inspired by a possible engine monitoring system [8]; it is clear that the topology of this example is a 5-node subsystem repeated 5 times. In total this network has 55 packet flows mapped to the 25 nodes; 25 of these flows are defined to be of HI-criticality and 30 of LO-criticality.

We will use the 5-node subsystem to illustrate the analysis associated with AirTight. The 5-node subsystem was also used in a prototype implementation of the protocol. This implementation[2]

[2]The prototype implementation is based on IEEE 802.15.4-compliant node hardware (the Iris XM-2110 nodes [22] manufactured by Crossbow Technology) and the TinyOS version 2 operating system [27] as obtained from the development repository in July 2017 [10].

is described in detail in a conference paper [8] and therefore is not covered here. In addition to the implementation, a protocol-accurate simulator for AirTight has been produced and validated against the prototype.

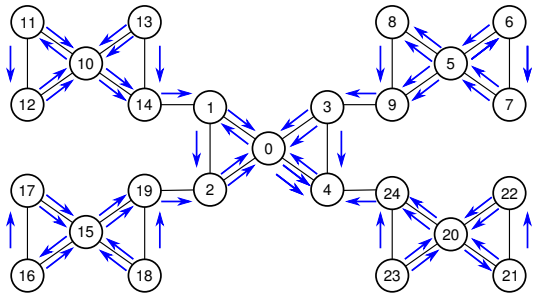


Fig. 2. Communication Graph of a 25 node Health Monitoring System

9.2 Example of Analysis

In this section we give an overview of single-channel slot table construction for an example 5 node network. Later we extend it to the multichannel case, showing how the provision of a second channel can provide more capacity without increasing the table size.

9.2.1 Single Channel Analysis. In this section we analyse a simple five node example which is motivated by the subsystem identified in the avionics use case (as illustrated in Figure 2). The 5 nodes (which are the central group in Figure 2) are depicted in Figure 3 and form a star topology: $n_1, n_2 \leftrightarrow n_0 \leftrightarrow n_3, n_4$. So n_0 can communicate directly with all nodes; but n_1, for example, can only communicate with n_2 and n_0, and not with n_3 or n_4. However, we assume conservatively that the interference graph of the system is complete such that all nodes may potentially interfere with each other, even though they may be out of range for intelligible communication. Therefore, we do not allow node n_1 to transmit to n_2 at the same time as n_3 transmits to n_4. We use periods and deadlines that are 1/5 of those of the larger example as this allows the tightness of the analysis to be illustrated. It also means that the full 25 node example is schedulable if this subsystem is.

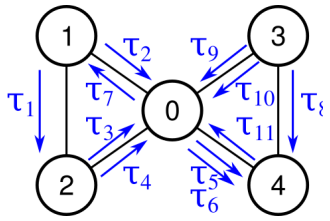


Fig. 3. Diagram of the nodes and communicating packet flows

There are nine end-to-end packet flows of two criticality types. Two packet flows must be routed through n_0; these are accommodated by simply, for this example of the analysis, dividing the deadline in two. This results in the eleven packet flows that are given in Table 1 (where P is the local priority, with 1 being the highest).

For this simple example we assume a single source of interference; the fault model is defined as follows:

(1) For $b = 5$ and $T^b = 100$, all deadlines must be met
(2) For $b = 15$ and $T^b = 100$, the deadlines of all HI-criticality flows must be met
(3) For $b > 15$ and any $T^b < 100$, best effort – send HI-criticality packets when possible, perhaps using a secondary parameter (importance) to order local access.

If we start with a table of length 5 with all nodes having a single slot then n_1, n_2, n_3, and n_4 are schedulable, but n_0 is not; with n_0 having two slots then all nodes are schedulable. Worst-case response times are given in Table 1. These invariably occur following faults of the worst possible magnitude (as defined by the fault model). Note that no assumptions have been made about where in the table any node's particular slot(s) are actually positioned. We use a simple formulation of the analysis, the supply function for all nodes apart from n_0 is 1 in 7, 2 in 13, 3 in 19 etc. For n_0 it is 2 in 7, 4 in 13, 6 in 19, 8 in 25, 10 in 31 and 12 in 37.

To give an example of the analysis; consider τ_5 which is the lowest priority packet flow on node n_0. It is a HI-criticality flow but first its worst-case response-time in the LO-criticality mode must be computed. Using equations (3) and (4) we initially have $X = 3$ and $R_5(LO) = 13$. Now equation (3) becomes:

$$X = 3 \ + F_5(LO, 13) + \left\lceil \frac{13}{26} \right\rceil 1 + \left\lceil \frac{13}{64} \right\rceil 1$$

$F_5(LO, 13)$ is 2, since one table is corrupted within which there are two slots, hence X becomes 7 and $R_5(LO) = 25$. At this point, iteration has converged, as the value of X does not change when 13 is replaced by 25.

To compute $R_5(HI)$ we can start with a value of 25 (since $R_i(HI) \geq R_i(LO)$) so equation (5) becomes:

$$X = 3 \ + F_5(HI, 25) + \left\lceil \frac{25}{26} \right\rceil 1 + \left\lceil \frac{25}{64} \right\rceil 1$$

The fault load, $F_5(HI, 25)$ is now 6 (three tables corrupted), so $X = 11$ and $R_5(HI) = 37$. Another iteration gives:

$$X = 3 \ + F_5(HI, 37) + \left\lceil \frac{25}{26} \right\rceil 1 + \left\lceil \frac{37}{64} \right\rceil 1$$

which is again 11; so $R_5(HI)$ has converged to 37. Note the interval for interference from the LO-criticality packet flow τ_6 is capped at 25, which is the response-time of τ_5 in LO-criticality mode (i.e. $R_5(LO)$).

(Note that a blackout of length 5 ($b = 5$) can only affect one table of length 5. This is because the table is a simple repeating static cycle that can be assumed to start at any point, including synchronised with the blackout).

9.2.2 Example of Table Construction for Multichannel Network. In a multichannel network it is possible to reduce the length of the slot table, by making use of parallelism in the network. For simplicity we will here concentrate on the introduction of a second channel to the example network. If the network was as shown in Figure 3 then there would be no benefit from multiple channels, since the network is sufficiently small that no two nodes can be transmitting simultaneously. For example, nodes 1 and 3 cannot be allocated slots on channels 0 and 1 simultaneously, since node 0 may be required to receive a packet from both simultaneously.

However, consider the modified scenario depicted in Figure 4. Two new nodes n_5 and n_6 are introduced into the network, together with flows τ_{12} and τ_{13}. If nodes n_5 and n_6 only require a single transmission slot, it is possible to allocate node n_5's transmission slot on the alternate channel in parallel with the slot of node n_1 on the original channel. Similarly, node n_6 can be allocated on the alternate channel in parallel with node n_2 on the original channel. Therefore the table length

Name	From	To	Criticality	T	D	C	P	R
τ_1	n_1	n_2	LO	30	30	2	2	25
τ_2	n_1	n_0	LO	26	13	1	1	13
τ_3	n_2	n_0	HI	40	40	1	2	31
τ_4	n_2	n_0	LO	13	13	1	1	13
τ_5	n_0	n_4	HI	38	38	3	3	37
τ_6	n_0	n_4	LO	26	13	1	1	13
τ_7	n_0	n_1	HI	64	32	1	2	31
τ_8	n_3	n_4	LO	32	14	1	1	13
τ_9	n_3	n_0	HI	64	32	1	2	31
τ_{10}	n_3	n_0	LO	32	32	2	3	31
τ_{11}	n_4	n_0	HI	40	40	2	1	31

Table 1. Example, Parameters and Response-Time Calculations For Single-Channel (SCA)

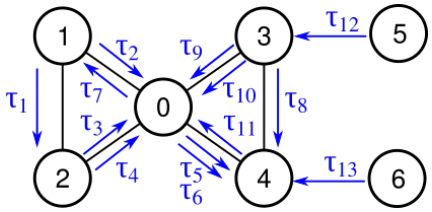


Fig. 4. Diagram of the nodes and communicating packet flows (multichannel alternative scenario)

would not increase over the original scenario depicted in Figure 3.

10 CONFIGURATION OPTIMISATION USING A GENETIC ALGORITHM

In Section 8, heuristics have been defined for a number of factors in the design space, specifically the slot table size, channel affinity sets, and routing from source to destination for multi-hop data. These provide a basic intuitive approach for configuring a system, but may not be optimal for a particular configuration. However, the schedulability problem is complex with a number of interactions. For example, adding additional transmission slots increases capacity available for a particular node, which could make a particular flow schedulable, but could also impact schedulability for another flow by lengthening the table. In addition, multichannel usage or modifications to channel affinity sets may reduce interference from higher priority flows, but it could also require additional transmitting and listening slots on another channel. These non-obvious interactions motivate the use of a search-based optimisation framework for configuration of the AirTight protocol.

10.1 Genetic Algorithm Structure

We follow a schedulability-driven GA model that has been successfully used in the optimisation of networked real-time systems [30]. In our implementation, the GA operates by evolving all population of chromosomes encoding slot table allocations. The initial population is created using the heuristics described in Section 8. Selected chromosomes are then modified via the operations of mutation and crossover in order to produce variations in the slot allocations. The quality of

n_0	n_1	n_2	n_3	n_4	n_5
0	3	2	0	1	1

Fig. 5. An example slot allocation count vector for a 6 node network with the single-channel GA. Elements give the number of slots for the node indicated in the header.

these allocations is evaluated by testing the schedulability according to the flowset equations in Sections 6.2 and 7. If the candidate solutions improve the fitness, then they are integrated into the population. The chromosomes with the lowest fitness are removed from the population. This process is repeatedly for a number of "generations" that results in a progressive refinement of the population towards successful schedulability. It terminates successfully when the population includes at least one slot allocation that enables full end-to-end schedulability of the flowset, or unsuccessfully if such allocation is not found over a customisable limit to the number of generations.

10.2 Single-Channel GA

Within a single channel AirTight system, the only consideration for a node is whether it is currently transmitting upon a particular slot. Given the assumption that all nodes are line powered, it is not necessary to optimise duty cycles and idle states, and thus all nodes which are not transmitting can be assumed to be listening on the single channel.

The GA for the single-channel system takes as input the number of network nodes and the flowset (which also informs it of which nodes are sink-only and which are active). We chose to structure chromosomes as an integer vector (commonly used in many GA implementations [28]), where each element represents the number of slots in the table allocated to a particular node (depicted in Figure 5). Sink nodes do not need any transmission slots, therefore it can be assumed that the allocation vector element representing them is zero. The table size can be computed as the sum of the allocation vector, and the kth element gives the value of a_k to be used in the analysis.

The initial population is as defined by the heuristic slot table generator but with some additional random variation in order to provide diversity in the population. Every node that is needed for transmission has the number of slots required by the heuristic plus an additional randomly selected number (which may be zero).

Evolution across generations is a result of mutation and crossover operations, which are random in nature. The mutation operation consists of varying elements of the slot allocation vector, by incrementing or decrementing the slot allocation. It is obviously unnecessary for nodes that only receive to have a transmission slot, and therefore vector values for these nodes remain at zero during mutation. During a mutation allocation M_C nodes with transmission flows are chosen to be altered, and for each node, the number of slots allocated is incremented by $\{-M_V, -M_V + 1 \ldots M_V\}$.

One-point crossover is performed upon two chromosomes G_1 and G_2 of length N by selecting a random crossover point in the table structure v and creating a new vector by combining the elements from $[G_1[0] \ldots G_1[v]]$ and $[G_2[v+1] \ldots G_2[N]]$.

The fitness function is computed by executing the analysis as described in Section 6.2, and returning a number indicating the proportion of flows which are schedulable within the flowset, for each level of criticality.

10.3 Multichannel GA

For a multichannel AirTight system, the scheduling and optimisation problem is more complex due to the requirement to consider receiver listening and correct channel tuning. A receiving node must be tuned correctly to the channel that its intended transmitter is using in the appropriate slot,

Channel	n_0	n_1	n_2	n_3	n_4	n_5
C_0	0	1	0	0	1	0
C_1	0	1	2	0	0	1
C_2	0	0	0	0	1	1

Fig. 6. An example slot allocation count vector for a 6-node network for the multichannel GA on 3 channels. Elements give the number of slots on the channel for the node indicated in the header

otherwise the transmission will fail. Therefore, the optimisation process must solve at least two partially independent but connected problems: firstly, determining the allocation vectors giving the number of slots to use on each channel, and secondly determining an optimal packing of these slots to meet channel activity constraints (for example that a node can only perform one action at a time). There is a third dependent problem, which is determining the affinity sets for the nodes. It is also possible for the GA to determine the affinity sets, which may improve performance either by reducing interference or enabling a smaller table size. This will be explored within the experimental section.

Like its single-channel counterpart, the GA for the multichannel system also takes as input the number of network nodes and the flowset. In addition, it requires channel affinity sets, which define the channels which a flow can use for transmission (as described in Section 7.1). From those inputs, it then builds active node peer sets (i.e. the sets of all receivers for a given transmitter node).

To accommodate the multichannel alocation problem, chromosomes are structured as a 2D array representing the number of slots allocated to the nodes over each specific channel, as illustrated in Figure 6. Rows represent the channels and the value in each column represents the number of slots allocated. Therefore, a particular element gives the integer count of all slots on the given channel for a particular node. Mutation is performed in a similar manner to the single channel GA, varying the table elements to increment and decrement the channel allocation count. One-point crossover is again applied by copying part of the node assignments from the source vector and another from the second.

The fitness function consists of the proportion of the flowset schedulable under the multichannel scheduling equations given in Section 7.2, for each level of criticality. The only complex operation is the computation of the table sizes, which is more complex than the single channel SCA approach. This is because the table sizing corresponding to a multichannel allocation array depends on the particular parallelism obtained. Bounds can be established such that the theoretical minimum table size for a channel allocation is the maximum sum of slots required on any channel. The theoretical maximum table size is the sum of allocations on all channels, and represents the case in which there is no transmission parallelism possible and the situation degenerates to a single channel used per slot (as in the small case study of Figure 3). To determine an exact table size corresponding to the current multichannel schedule vector, schedule packing is performed as described in Section 10.3.1.

10.3.1 Schedule Packing. In the schedule packing problem, the problem is to determine a packing for the schedule allocations required which minimises the table size while meeting all the following constraints SPC_n:

- SPC_1: A node must only perform a single action on a single channel during a slot. This is due to the assumption of nodes only having a single channel radio.
- SPC_2: All potential receivers of a node must be tuned correctly to receive from it during its transmission slot(s). For example, if node n_1 transmits data flows to receivers n_3 and

n_4, then both nodes n_3 and n_4 must be listening during all of node n_1's TX slots; it is not possible to reassign node n_4 to another channel. This is because node n_1 must be free under the AirTight protocol to make its local decision as to which packet to send.

- SPC_3: A node may not be allocated so as to receive from more than one transmitter simultaneously, even if they are on the same channel. This is because transmission decisions are decentralised, and two nodes may decide to transmit at the same time, which would cause a collision at the receiver.
- SPC_4: In this paper we make an additional assumption: the single-domain constraint. This is an assumption that the interference graph for the topology is complete, so there can only be one transmitter active on a particular channel during any slot. This is similar to the assumption conventionally employed in industrial applications of WirelessHART [17]. This is a stronger condition than SPC_3.

The algorithm that constructs the schedule table proceeds as follows:
Inputs: (I1) The slot allocation count vector, illustrated in Figure 6, which gives the number of slots required for each node-channel combination. (I2) The set of potential receivers for every transmitting node-channel combination $RCV(TX_n, Ch)$. This is determined from the set of flows and their channel affinity sets.
Outputs: (O1) The schedule table.
Intermediate: A channel tracking vector is used to track the use of channels in the current slot. All channels are set initially to free.

1. The current slot $s = 0$, representing operation on the first slot of the schedule table. The schedule table is initially empty.
2. All channels in the channel tracking vector are set to free.
3. The slot allocation count vector is scanned for an element (node TX_n, channel Ch) with a non-zero value that meets the following constraints: (i) the node TX_n is free in slot s of the schedule table; (ii) all of the receiving nodes required (i.e. in the set $RCV(TX_n, Ch)$) are also free in slot s of the schedule table; (iii) channel Ch is free in the channel tracking vector.
4. If no element is found that meets the above constraints then $s = s + 1$ and control returns to step 2. Otherwise, for slot s in the schedule table, node TX_n is set to transmitting on channel Ch, and all its potential recipients in the set $RCV(TX_n, Ch)$ are set to listening (to TX_n) on channel Ch. Further, the entry in the slot allocation count vector for node TX_n channel Ch is decremented, and channel Ch is set to used in the channel tracking vector.
5. If all elements of the slot allocation count vector are zero, the algorithm terminates, with $s + 1$ as the size of the slot table. Otherwise control returns to step 3.

Note that the algorithm terminates since it adds slots to the table until all of the entries in the slot allocation count vector have been decremented to zero. (Each time a slot is added, at least one entry is decremented).

10.4 GA Optimisation Of Affinity Sets And Slot Tables

The heuristic for constructing affinity sets was defined in Section 8.2. However, it may be possible to improve performance by allowing the GA to mutate affinity sets during its optimisation process.

The overall approach in this case is based on the multichannel GA described in Section 10.3, with the following modifications:

- The genome is structured as a pair containing both the slot table allocation array and the affinity sets. An initial population is defined which begins with both of these calculated using the heuristics defined in Section 8.

- During the mutation operation, the initial population is modified by altering either one of the slot table or the affinity sets. Mutation of the slot table is as described in the general multichannel GA in Section 10.3. Mutation of the channel affinity sets consists of selecting a flow and randomly either adding or removing a particular channel from its affinity set.

11 EVALUATION

This section contains experimental results showing the performance of the various analytical schedulability equations on tables constructed via the set of heuristics, and the results of improvement of schedulability using genetic algorithms.

11.1 Evaluating Heuristic-based Configurations

In order to test the schedulability properties of the various analysis techniques independently of GA performance, the heuristic is used to build slot tables that serve as a baseline. This will lead to some flows not being schedulable, since it may construct larger tables that lead to deadline misses during the iterative analysis algorithm. However, it provides capacity to each node based upon their loadings, as described in Section 8.1. For MCA, channel affinities are assigned as described in Section 8.2. Then, in order to exploit parallelism inherent in this flowset, the schedule packing is performed using the algorithm described in Section 10.3.1.

The experimental procedure in this section generates flowsets, according to the distribution parameters given in Table 2. Four different experimental configurations are used in the generation of these flowsets, configurations A to D. These configurations have various values for the proportion of HI criticality flows, and, in the analysis used, different values for the length of faults, and the minimal spacing time before the start of subsequent faults. The values of these parameters are listed in the table.

Typically a flowset gives rise to a median value of three single-hop flows once routing is taken into account. For example, over 1000 generated flowsets, with 30 end-to-end flows, the minimal number of flows obtained was 63, a median of 87 and a maximal value of 105. The schedulability of the flowsets is tested according to each of the algorithms presented in Section 6.2 and Section 7: SCA and MCA. A flowset is considered schedulable if all its constituent end-to-end flows are schedulable. If any flow within it is not schedulable, then the entire flowset is not schedulable. Under SCA and MCA, individual hops are not required to meet the sub-deadline assigned to the hop, as long as the final worst-case response time at the destination is less than the flow end-to-end deadline. Figure 7 illustrates the schedulability of flowsets of increasing size for SCA and MCA. Several scenarios are tested, with faults of varying lengths. The long and short fault parameters are as described in Table 2.

With MCA, the increasing number of channels available in the network improves the performance over SCA by increasing the overall bandwidth available and permitting simultaneous transmissions, which leads to better schedulability at higher loadings. However, the optimal number of channels is actually 2 or 3, with 4 channels producing a reduction in schedulability. This is as expected and is due to the interaction between affinity set heuristic and schedule generation, specifically additional slots are required for the support of the additional channels which cannot be parallelised, and this negates the advantage from the reduction of the slot table sizes or separation of interference sets. With the case in which faults are more frequent, in Figure 7c, the overall schedulability is lower but separation between the various MCA channel variants is lower.

Parameter	Value
Node count	36 (6x6 topology)
Standard flow count in flowset	40
HI criticality proportion (experiment configurations A, B, C)	25%
LO criticality proportion (experiment configuration D)	50%
Fault length (experiment configurations A, D)	10 (LO), 30 (HI)
Fault length (experiment configurations B, C)	5 (LO), 15 (HI)
Minimum time between fault starts (experiment configurations A,B,D)	100 slots
Minimum time between fault starts (experiment configuration C)	75 slots
Minimum period of flows	200
Maximum period of flows	1000
Flowsets tested in heuristic experiment (Section 11.1)	10000
Flowsets tested in GA experiment (Section 11.2)	1000

Table 2. Default flowset parameters for experiments

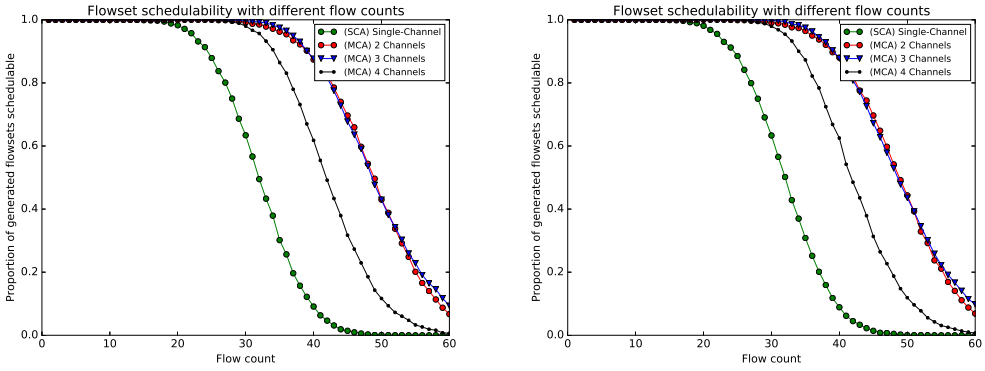
11.2 GA-based Optimisation of Slot Table Construction

This section considers the performance of the genetic algorithms described in Section 10 for generating slot tables for the AirTight protocols in the single channel (SCA) and multichannel analysis (MCA) cases. In the single-channel case, the progression of the GA in terms of evolution towards schedulability is presented in Figures 8a and 8b. (Note these figures are best viewed online in colour). The line plots show the number of generations taken for flowsets to achieve schedulability in both LO and HI crit modes, with the generation number listed on the horizontal axis and the schedulability proportion at this generation on the y axis. Values between 0 to 0.5 indicate the proportion of flows in the flowset that are schedulable in the LO mode, and values from 0.5 to 1.0 indicate the proportion schedulable considering HI mode.

In the SCA case (Figure 8a), a number of flowsets require a significant number of generations to reach schedulability, and around 9% had not attained schedulability by the time the experiment ended at 500 generations (some requiring additional slots for schedulability in the HI mode). In the MCA example (Figure 8b) a majority of the flowset cases become schedulable quickly within a few generations, since the multi-channel and schedule packing process enables schedulability more easily by permitting a smaller slot table, and the scheduling problem is threfore easier. However, a few clear outliers take longer to achieve schedulability.

The number of generations of the GA necessary to achieve final schedulability of a flowset may also be represented as a histogram, in order to more clearly see the behaviour of both approaches at a low number of generations. This is illustrated in Figures 8c and 8d. These histograms include flowsets which attained schedulability below 100. SCA has a much broader spread of generations to reach schedulability, and in comparison MCA is likely to become schedulable immediately or with minor modification in a single generation or two after the evolution begins (reflecting that the slot table heuristic is relatively good at this table size). Fewer flowsets require more than 20 generations for MCA. There are a small number of outliers clustered under the 20-25 generation range for MCA.

11.2.1 Variations In Fault Length. Altering the fault lengths by making the fault model accommodate longer or shorter faults can change the schedulability and therefore the fitness function values delivered which impact GA performance. In particular, shorter absolute faults could lead to



(a) Configuration A - Long faults

(b) Configuration B - Short faults

(c) Configuration C - Frequent faults

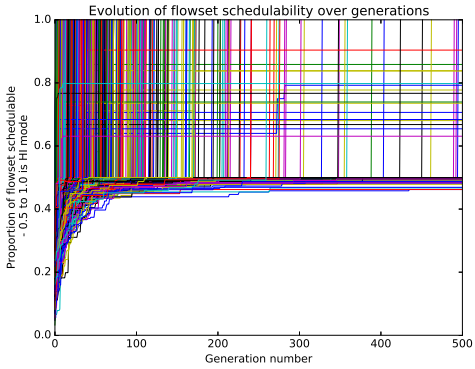
(d) Configuration D - Long faults (50% HI)

Fig. 7. Schedulability - SCA and MCA for varying numbers of channels with different fault scenarios

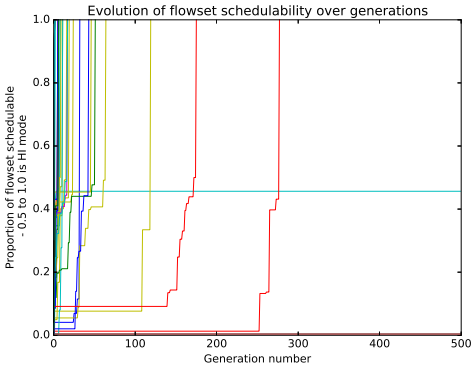
smaller slot tables, since there would be less cascading impact from higher priority flows to lower priority flows.

However, make the fault repeat interval more frequent would potentially increase the number of times the analysis exceeds a fault boundary, and therefore take longer to reach schedulability. In particular, when the fault periodicity was halved and the fault length (in both LO and HI modes) halved a majority of the scenarios became unschedulable even at the end of the GA process. Therefore, the fault length was reduced by half in this experiment while retaining the constant fault periodicity.

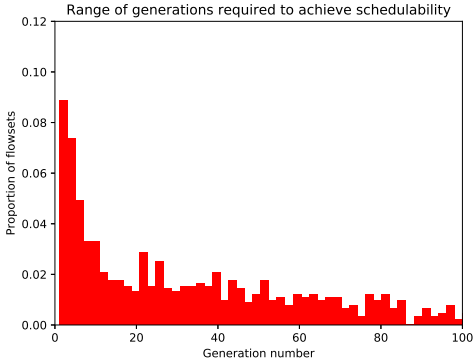
Figures 9a, 9b, 9c and 9d illustrate the corresponding results for these shorter fault cases. (Again, these figures are best viewed online in colour). They show that the shorter fault cases are broadly similar to the longer fault case assessed previously in Figure 8; however, one notable difference is that for SCA scheduling lines (Figure 9a), the addition of slots during GA evolution is normally sufficient to make the flowset entirely schedulable in HI mode as well as LO. This accounts for the frequent near vertical steps from 0.5 to 1.0 in a number of flowsets approaching complete



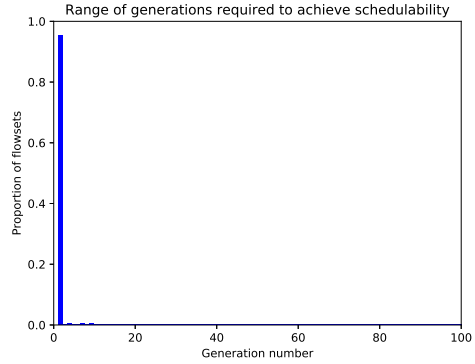
(a) Flowset schedulability improvement for SCA with generation number



(b) Flowset schedulability - MCA with 3 channels



(c) Flowset schedulability histograms - SCA



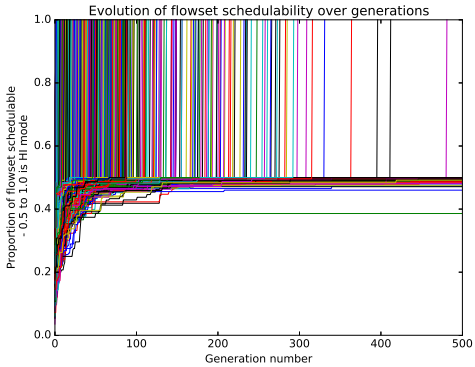
(d) Flowset schedulability histogram - MCA with 3 channels

Fig. 8. GA Evolution Progress With Standard Faults

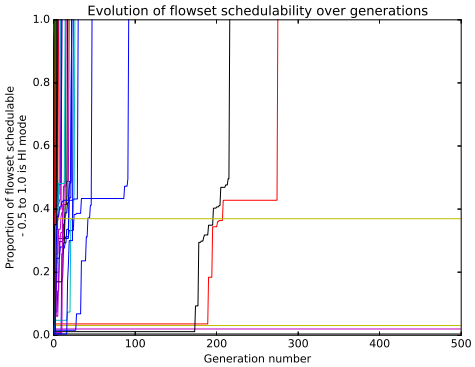
schedulability. perhaps reflecting the less challenging fault conditions. The shape of the histograms and the associated outliers is also broadly similar to the previous experiment.

11.3 GA Optimisation of Affinity Sets And Slot Table Structure

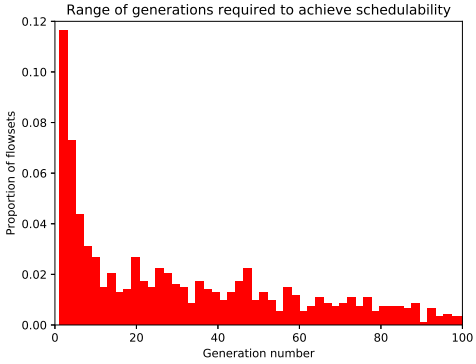
With multichannel analysis it can be is possible to mutate the affinity sets during the evolutionary process, as described in Section 10.4. In this case, the channel affinity sets are modified within the evolutionary process, giving more choice in the allocation decisions. Figures 10a and 10b show the equivalent schedulability progression and histograms for the standard fault case. Overall the time taken for evolution and the progress towards schedulability is similar to that without affinity set modification, although few outliers tend to reach schedulability more quickly (i.e. there are no outliers in the over 200 range). Also, some flowset/affinity set combinations do not progress towards schedulability under this GA. This may be because the larger search space produced by affinity set modification does not permit any improvements to the scheduling problem in the given



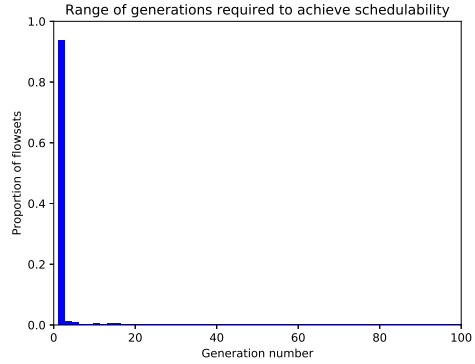
(a) Flowset schedulability improvement for SCA with generation number



(b) Flowset schedulability - MCA with 3 channels



(c) Flowset schedulability histograms - SCA



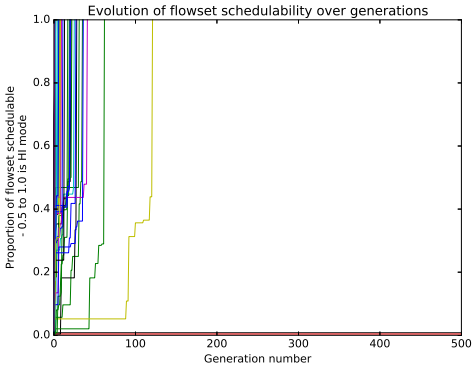
(d) Flowset schedulability histogram - MCA with 3 channels

Fig. 9. GA Evolution Progress With Shorter Faults

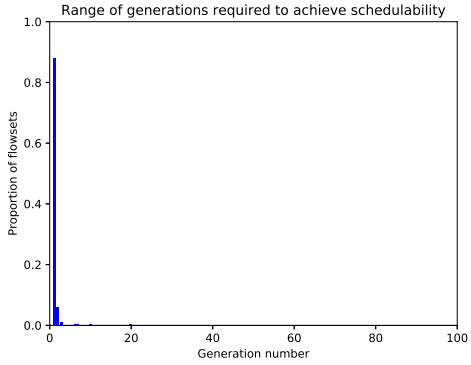
number of generations, or because modifications to the affinity sets interfere with optimisations on slot tables.

11.4 Schedulability Improvement From the GA At Different Flow Counts

In Section 11.1 the results considered schedulability resulting from the starting heuristics for slot table and affinity set construction. This section considers the final schedulability improvements delivered by the genetic algorithm. Since this is produced by evolutionary improvement of an original population, this allows the relative benefit of the GA to be assessed for different flowset sizes. In Figure 11 the improvements generated by GA evolution are presented for the long and short fault cases, represented by the gap between the original and GA improved lines. The GA improved case is only evaluated at increments of 5 flows, due to the time taken to execute it. Since 3 channels typically performed best in 7a, 3 channels are used for MCA. It is notable that in Figure 11a the GA improvement is lower for MCA than SCA at flow counts less than around 45, since

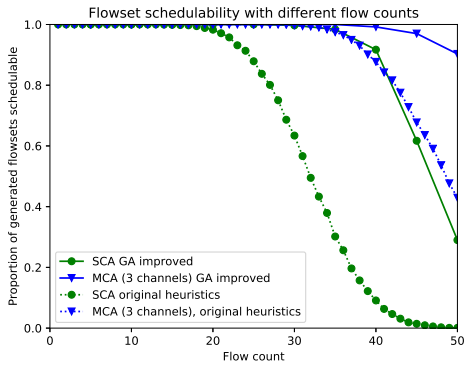


(a) Flowset schedulability - MCA with 3 channels

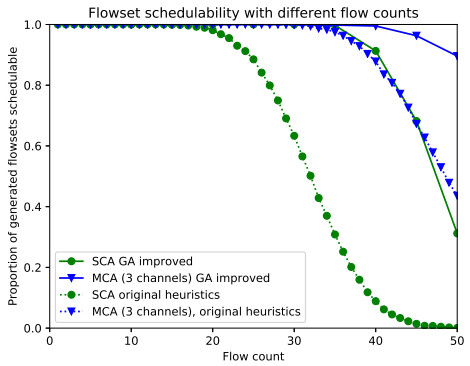


(b) Flowset schedulability histogram for short faults - MCA with 3 channels

Fig. 10. GA Evolution Progress With Channel Affinity Set Mutation



(a) Long fault case



(b) Short fault case

Fig. 11. Schedulability curves before and after GA improvement

MCA with heuristics is already starting at a higher value. At 40 flows, the case used in the earlier experiments, the SCA GA was able to raise the schedulable proportion from around 8% (original slot table heuristics) to nearly 90% after 500 generations of the GA. However, its performance declines at 50 flows, representing a significantly more challenging scheduling situation. For MCA, the GA is able to achieve almost 90% of flows schedulable at the end of 500 generations.

12 CONCLUSION AND FUTURE WORK

This paper presents schedulability analysis for single-channel and multichannel multi-hop wireless-enabled Cyber-Physical Systems based on the AirTight protocol. Heuristics are specified and evaluated as a basis for the starting point of design space exploration demonstrating the schedulability performance for AirTight's slot table development. Genetic algorithms are then defined and

evaluated to assess their performance in developing schedule tables incorporating multichannel allocation in these systems. In future work we aim to analyse the impact of adaptive routing (since routing was kept constant using the shortest path available during this work) and the extent to which routing can be customised with respect to criticality (perhaps to allow nodes more options for HI criticality traffic in the presence of faults).

Acknowledgements

The research described in this paper is funded, in part, by the EPSRC grants MCCps (EP/P003664/1). No new primary data was created during this study.

REFERENCES

[1] A. Alemayehu, L. George, V. Sciandra, and M. Agueh. Mixed criticality scheduling applied to JPEG2000 video streaming over wireless multimedia sensor networks. In *Proc. of the Workshop on Mixed Criticality (WMC 2013)*, pages 55–60, 2013.

[2] N.C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[3] N.C. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[4] S.K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 34–43, 2011.

[5] A. Burns and R.I. Davis. Mixed criticality on controller area network. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, pages 125–134, 2013.

[6] A. Burns and R.I. Davis. A survey of research into mixed criticality systems. *ACM Computer Surveys*, 50(6):1–37, 2017.

[7] A. Burns, J. Harbin, and L.S. Indrusiak. A Wormhole NoC protocol for mixed criticality systems. In *Proc. IEEE Real-Time Systems Symposium*, pages 184–195. IEEE, 2014.

[8] A. Burns, J. Harbin, L.S. Indrusiak, I. Bate, R.I. Davis, and D. Griffin. AirTight – A resilient wireless communication protocol for mixed-criticality systems. In *Proc. RTCSA*, 2018.

[9] A. Burns, S. Punnekkat, L. Strigini, and D.R. Wright. Probabilistic scheduling guarantees for fault-tolerant real-time systems. In *Proc. of the 7th International Working Conference on Dependable Computing for Critical Applications. San Jose, California*, pages 339–356, 1999.

[10] TinyOS core developers. TinyOS development repository. https://github.com/tinyos/tinyos-main, 2017. Master branch as of July 2017.

[11] R.I. Davis and N. Navet. Traffic shaping to reduce jitter in controller area network (CAN). *SIGBED Review*, 9(4):37–40, 2012.

[12] A.C. Dimopoulos, G. Bravos, G. Dimitrakopoulos, M. Nikolaidou, V. Nikolopoulos, and D. Anagnostopoulos. A multi-core context-aware management architecture for mixed-criticality smart building applications. In *11th System of Systems Engineering Conference (SoSE)*, pages 1–6, 2016.

[13] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proc. of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 1–14, 2012.

[14] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proc. Information Processing in Sensor Networks (IPSN)*, pages 73–84, 2011.

[15] T. Fleming and A. Burns. Extending mixed criticality scheduling. In *Proc. 1st Workshop on Mixed Criticality Systems (WMC), RTSS*, pages 7–12, 2013.

[16] HART Communications Foundation. Document No 7.5: HART communication protocol specification. Technical report, HARTComm, 2013.

[17] HART Communications Foundation. Iec 62591: Industrial networks - wireless communication network and communication profiles - WirelessHART (tm). Technical report, IEC Geneva, 2016.

[18] S. Gandham, M. Dawande, and R. Prakash. Link scheduling in wireless sensor networks: Distributed edge-coloring revisited. *Journal of Parallel and Distributed Computing*, 68(8):1122–1134, 2008.

[19] A. Gujarati, F. Cerqueira, and B. Brandenburg. Multiprocessor real-time scheduling with arbitrary processor affinities: from practice to theory. *Real-Time Systems*, 51(4):440–483, Jul 2015.

[20] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. In *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 3–12, April 2011.

[21] J. Harbin, D. Griffin, A. Burns, I. Bate, R. Davis, and L. Indrusiak. Supporting critical modes in AirTight. In *Proc. 6th Int. Workshop On Mixed Criticality Systems (WMC)*, 2018.

[22] MEMSIC Inc. IRIS wireless measurement system. www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf, 2011. Original document from Crossbow Technologies.
[23] L.S. Indrusiak, J. Harbin, and A. Burns. Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip. In *Proc. European/Euromicro Conference on Real-Time Systems (ECRTS)*, pages 47–56. IEEE, 2015.
[24] X. Jin, J. Wang, and P. Zeng. End-to-end delay analysis for mixed-criticality WirelessHART networks. *IEEE/CAA Journal of Automatica Sinica*, 2(3):282–289, 2015.
[25] X. Jin, C. Xia, J. Wang, and P. Zeng. Mixed criticality scheduling for industrial wireless sensor networks. *Sensors*, 16(9):1376, 2016.
[26] G. Kunzel, G.P. Cainelli, I. Muller, and C.E. Pereira. Weight adjustments in a routing algorithm for wireless sensor and actuator networks using Q-learning. *IFAC-PapersOnLine*, 51(10):58 – 63, 2018. 3rd IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control CESCIT 2018.
[27] P. Levis and D. Gay. *TinyOS Programming*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
[28] S. Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. Available for free at http://cs.gmu.edu/~sean/book/metaheuristics/.
[29] R. Mangharam, A. Rowe, R. Rajkumar, and R. Suzuki. Voice over sensor networks. In *Proc. 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pages 291–302, 2006.
[30] P. Mesidis and L. S. Indrusiak. Genetic mapping of hard real-time applications onto NoC-based MPSoCs – a first approach. In *Proc. 6th Int Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, 2011.
[31] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.
[32] A. Rowe, R. Mangharam, and R. Rajkumar. RT-Link: A time-synchronized link protocol for energy- constrained multi-hop wireless networks. In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, volume 2, pages 402–411, 2006.
[33] A. Saifullah. *Real-time wireless sensor-actuator network for cyber-physical systems*. PhD thesis, Department of Computer Science and Engineering, Washington University in St Louis, 2013.
[34] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen. Schedulability analysis under graph routing in WirelessHART networks. In *IEEE Real-Time Systems Symposium*, pages 165–174, 2015.
[35] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. Real-time scheduling for WirelessHART networks. In *2010 31st IEEE Real-Time Systems Symposium*, pages 150–159, 2010.
[36] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. End-to-end delay analysis for fixed priority scheduling in WirelessHART networks. In *17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 13–22, 2011.
[37] W. Shen, T. Zhang, F. Barac, and M. Gidlund. PriorityMAC: A priority-enhanced MAC protocol for critical traffic in industrial wireless sensor and actuator networks. *IEEE Transactions on Industrial Informatics*, 10(1):824–835, 2014.
[38] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, pages 239–243, 2007.
[39] C. Xia, X. Jin, L. Kong, J. Wang, and P. Zeng. Transmission scheduling for mixed-critical multi-user multiple-input and multiple-output industrial cyber-physical systems. *International Journal of Distributed Sensor Networks*, 13(12), 2017.
[40] C. Xia, X. Jin, L. Kong, and P. Zeng. Bounding the demand of mixed-criticality industrial wireless sensor networks. *IEEE Access*, 5:7505–7516, 2017.
[41] C. Xia, X. Jin, L. Kong, and P. Zeng. Scheduling for emergency tasks in industrial wireless sensor networks. *Sensors*, 17(7), 2017.
[42] Q. Zhang, F. Li, L. Ju, Z. Jia, and Z. Zhang. Reliable and energy efficient routing algorithm for WirelessHART. In Xian-he Sun, Wenyu Qu, Ivan Stojmenovic, Wanlei Zhou, Zhiyang Li, Hua Guo, Geyong Min, Tingting Yang, Yulei Wu, and Lei Liu, editors, *Algorithms and Architectures for Parallel Processing*, pages 192–203, Cham, 2014. Springer International Publishing.
[43] X. Zhao, H. Gao, G. Zhang, B. Ayhan, F. Yan, C. Kwan, and J.L Rose. Active health monitoring of an aircraft wing with embedded piezoelectric sensor/actuator network: I. defect detection, localization and growth monitoring. *Smart Materials and Structures*, 16(4):1208, 2007.
[44] M Zimmerling, L Mottola, P Kumar, F Ferrari, and L. Thiele. Adaptive real-time communication for wireless cyber-physical systems. Technical report, ETH Zurich, 2016.

Received July 2019